Studying the Change Histories of Stack Overflow and GitHub Snippets

Saraj Singh Manes
School of Computer Science
Carleton University
Ottawa, Canada
saraj.manes@carleton.ca

Olga Baysal
School of Computer Science
Carleton University
Ottawa, Canada
olga.baysal@carleton.ca

***Abstract*—Stack Overflow is a popular Q&A forum for software developers, providing a large number of copyable code snippets. While GitHub is a collaborative development platform, developers often reuse Stack Overflow code in their GitHub projects. These snippets get revised or edited on each platform. In this work, we study Stack Overflow posts and the code snippets that are reused from these posts in GitHub projects. We investigate and compare the change history of SO snippets with the change history of GitHub snippets. We have applied a stratified random sampling when mining 440,000 GitHub projects to create a dataset representing the change history of the reused snippets; this dataset contains 22,900 GitHub projects, 33,765 Stack Overflow references mapped to 4,634 Stack Overflow posts, and a total of 73,322 commits.**

We analyze the evolution patterns of snippets on each platform, compare key trends and explore the co-change of these snippets. Our results demonstrate that 76% of snippets evolve on Stack Overflow, while only 22% of the reused code snippets evolve in GitHub. Stack Overflow snippets undergo fewer and smaller changes compared to their evolving counterparts on GitHub. The evolution of snippets on both platforms is driven by the original author of the content. Finally, we found that a small percentage of snippets is co-changing across two platforms, while snippets in GitHub and Stack Overflow evolve independently of one another.

***Index Terms*—Code snippets, change history, evolution, Stack Overflow, GitHub, time series, co-change, code reuse**

I. Introduction

Stack Overflow (SO) [1] consists of a large corpus of software development knowledge in the form of Question & Answer (Q&A) posts and extensive commentary on posted knowledge in the form of comments. This volume of information on various subjects is available in the form of posts that include text and code snippets. A code snippet is a small block of code in a particular choice of programming language. The knowledge, represented by text and accompanied by optional code snippets, serves as a description for a question or answer, asked by a registered user of the Stack Overflow platform. Stack Overflow consists of over 40 million such question-answer posts and the number keeps on increasing daily.

This vast information attracts a broad audience of software developers to become contributors or active consumers of the present knowledge. According to the official statistics of Stack Overflow [2], around 50 million developers visit Stack Overflow monthly to learn and share their knowledge. This large community of software experts with different levels of experience, participate not only to generate knowledge in the form of questions and answer posts but also help to validate created knowledge with upvotes and comments. SO users continuously monitor newly created threads and help authors of new posts in identifying weaknesses in their answers by submitting their commentary in the comment section of the post.

Dependence on Stack Overflow is so severe that even new learners such as students prefer to learn new programming languages and frameworks from the forum instead of diving into textbooks and learning via a traditional setting [3]. Ready-to-use code snippets provide an easy way to find solutions for daily programming problems that developers around the world face. Naturally, this knowledge collected in Stack Overflow is transferred to other software artifacts. This work studies one such migration of knowledge and its relation to a particular type of software artifacts, i.e., open-source projects hosted on GitHub. GitHub [4] is a `git`-based software repository hosting service that allows developers to collaborate on their project development. According to the SO licensing terms [5], developers who use code from Stack Overflow are required to attribute the code (e.g., by providing a URL as a comment in their code). Providing a reference to the knowledge source may also increase chances of pull request acceptance in an open-source project. Thus, a large number of references to Stack Overflow posts are found in GitHub projects [6].

While code reuse and knowledge sharing are effective practices that SO facilitates, recent studies have shown that code snippets can be toxic [7] and lead to license violations [8] or migration of security vulnerabilities [9], [10]. Therefore, understanding how SO snippets are reused in software projects is critical for mitigating the migration of bugs and vulnerabilities to software applications. First, we need to understand how SO snippets evolve over time compared to other software artifacts such as code. Initial analysis of the SO post evolution demonstrates that it differs from the evolution of software projects [6], [11]. To the best of our knowledge, no study has compared the SO snippet evolution with the evolution of the reused code snippets in GitHub. We are interested in investigating the patterns of SO and GH snippet evolution, and the possibility of the co-evolution of these snippets. The knowledge of snippet evolution may insight new research directions and tool support. For example, a tool that tracks SO snippet changes (and possibly checks for vulnerability in snippets [10]) can be integrated with CI tools to prevent bugs/vulnerabilities from being deployed into production.

Our research focuses on studying the evolution of SO snippets and their corresponding reused code snippets in GitHub to support a better understanding of code or content reuse across two collaborative development platforms. We define SO snippet as the answer (accepted if available, otherwise a highly voted one) that developers are referring to in their projects' code. SO snippets may include code snippet, pseudo code, algorithms, or only discussions that developers refer to. The aim of this work is to conduct a large-scale study on investigating the change and co-change patterns of snippets on Stack Overflow and GitHub. In this work, we address the following research questions:

- **RQ1: To what extent do snippets evolve in GitHub and Stack Overflow?**
  This question is related to gaining insights into the extent of the changes that snippets undergo on each platform.
- **RQ2: How and how often do snippets evolve in GitHub and Stack Overflow?**
  The answer to this question would quantify the snippet evolution in terms of the change size, type, and time.
- **RQ3: Who is driving the evolution of snippets on both platforms?**
  Results of this RQ would inform us about the ownership of the content on both platforms and the role of the original author in the evolution of snippets.
- **RQ4: To what extent do Stack Overflow snippets and the reused code snippets co-change?**
  If co-change of snippets across two different platforms happens, developers who reuse SO content need to be aware of the evolving SO content to be able to mitigate the effects of inaccurate information migration.

The contributions of this research work are as follows:

1) Supporting better understanding of the change history of snippets on two different collaborative platforms.
2) The change history dataset[1] for the reused code snippets in GitHub that can be further mined for conducting code clone detection and analysis.

II. Motivation and Example

Recently, Baltes et al. have released the SOTorrent dataset [12]. SOTorrent contains the version history of all SO posts created from the official data dump. Baltes et al. have performed an initial analysis [11] on Stack Overflow posts investigating the frequency and distribution of edits with respect to the creation time. One of the key results of their study is the evidence of the change as 35.9% of posts were

TABLE I: Example: a Stack Overflow snippet, no changes.

R.	**Content of a SO snippet**[2]
1	Here is some code you can use to query the worksheets from my copy of your spreadsheet.
	`#!/usr/bin/python`
	`from gdata.spreadsheet.service import SpreadsheetsService`
	`key = '0Aip8Kl9b7wdidFBzRGpEhoUlVPaEg2X0F2YWtwYkE'`
	`client = SpreadsheetsService()`
	`feed = client.GetWorksheetsFeed(key, visibility='public', projection='basic')`
	`for sheet in feed.entry:`
	`  print sheet.title.text`
	** Tips ** I find it really helpful when working with t ...

TABLE II: Example: the reused code snippet in the GitHub project with two revisions (code is formatted to fit the table).

R.	**Content of the reused code snippet in GH** [3]
1	`def load_google_spreadsheet_mapping_file(spreadsheet_key, worksheet_name=None):`
	`  """Some of this code is based on the following websites, as well as the gdata.spreadsheet.text_db module: http://stackoverflow.com/a/12031835`
	`  """`
	`  # TODO test if no connection`
	`  gd_client = SpreadsheetsService()`
	`  worksheets_feed = gd_client.GetWorksheetsFeed(spreadsheet_key, visibility='public', projection='basic')`
	`  if len(worksheets_feed.entry) < 1:`
	`    raise RemoteMappingFileError("The Google Spreadsheet with key "%s" does not have any worksheets associated with it." % spreadsheet_key)`
2	`try:`
	`  worksheets_feed = gd_client.GetWorksheetsFeed(spreadsheet_key, visibility='public', projection='basic')`
	`except gaierror:`
	`    raise RemoteMappingFileError("Could not establish connection with server. Do you have an active Internet connection?")`

edited at least once. The work of Baltes et al. has inspired us to study the evolution of snippets and whether it exhibits similar trends on Stack Overflow and GitHub. In different research fields, term "evolution" holds different meaning; in our work, we define *evolution* as the change history of snippets and use terms *change* and *evolution* interchangeably.

To demonstrate evidence of the evolution of snippets on Stack Overflow and GitHub, we provide an example of code

[1]https://github.com/manessaraj/GHCodeSnippetHistory

[2]https://stackoverflow.com/questions/7561148/retrieve-data-from-public-google-spreadsheet-using-gdata-library?answertab=active#tab-top

[3]https://github.com/biocore/qiime/blob/master/qiime/remote.py

reuse from Stack Overflow post to a real GitHub project. Table I presents a Stack Overflow post with a code snippet. This snippet has no changes after its creation in a Stack Overflow answer post. Table II illustrates how this SO snippet was reused in the GitHub project. This reused code snippet has been revised twice in the project's history. For brevity, we show only one revision of the reused snippet. We can also see the original version of the SO snippet and the attribution of the code (i.e., a URL to the SO post) in its comments. This reference to the original source of information in SO has allowed Baltes et al. [11] to build a dataset that links the SO content history with the GH projects. Our work extends this linkage with the change history of the reused code snippets than are integrated in the GitHub projects.

III. Related Work

A. Code Reuse

As a software system evolves, structural and behavioural changes become more complex and time-consuming. As such, code reuse from the previous version or third party vendors has become an interesting problem [13]. Code reuse is defined as "*the use of existing software or software knowledge to construct new software*" [14]. Researchers have developed different techniques on how to reuse existing code. Wang et al. [15] have outlined general principles for leveraging an existing code base while developing new features yet reducing code redundancy.

Code reuse can happen not just from other projects' codebases but also from online Q&A forums for developers such as Stack Overflow. Several researchers have studied the code reuse from Stack Overflow posts to GitHub projects [14] [16] [17]. Lotter et al. [14] have investigated the reuse of SO content in popular open-source Java projects. The authors have analyzed around 150K of Stack Overflow posts and detected their clones and reuse in 12 GitHub Java projects. They have also tried to determine clones from SO to GitHub projects, as well as clones among the GitHub projects. Clone results were 3.3% and 77.2%, respectively. While the former represents the code reuse from Stack Overflow to GitHub, the latter indicates that the reuse of code between GitHub projects is much more significant. While the results of their analysis were substantial, the number of projects under consideration was low.

Kamiya et al. [18] argue that code reuse leads to maintainability issues that could introduce multiple points of failure if reused code snippets are "buggy" and constitute to nearly half of the entire code snippets in their dataset.

Code reuse becomes more challenging if code is taken and reused from Stack Overflow due to potential code incompatibility and non-efficient solutions that can be tied to the lack of developer understanding of the context. Further, such snippet is well explained on the forum with a textual description of logic/rationale, yet it is up to the developer to figure out how to adapt this snippet, sometimes in different programming language, to the existing project effectively without introducing bugs. Ragkhitwetsagul et al. [19] argue that more and more developers tend to refer to Stack Overflow, while they do not understand certain aspects of code failure. Such lack of understanding when adapting the content from online forums affects the quality of the reused code.

Abdalkareem et al. [20] analyzed code reuse from Stack Overflow in mobile applications and found that around 1.3 % of the apps in their dataset were composed of code snippets from Stack Overflow. Further, the code borrowed and adapted from SO was introduced to the established apps later in their lifetimes.

Empirically, there is little evidence on how information is being adapted from Stack Overflow to GitHub. Yang et al. [21] investigated the impact of one platform on the other. The authors studied code snippets that exist on both platforms and used cloning and measure of similarity as a measure of equality of information present on both sides. The authors focused their work on Python GitHub projects. They found that 86% of code snippets on GitHub were reused copies of the rest of the 14%, indicating a large amount of code duplication, while such duplication on Stack Overflow was only 1.3%. Further, only 1% of code snippets were the same on both platforms. These findings demonstrate that clone studies have not been able to offer deeper insights into how code snippets are adapted from SO to GH projects.

Similar to Yang et al. [21], several researchers agreed that clones lead to bug propagation and software maintenance issues [18], while others argued that clones are not harmful at all, rather beneficial [22], [23]. Ragkhitwetsagul et al. [7] argued that there are two directions of snippet migration: 1) snippet is cloned from a software project to a Q&A website as an example; or 2) code is cloned from a Q&A website to a software project to perform some task. Yet, clones may lead to license violations. The authors investigated the toxic effects of such clones. Unlike other work on this topic, they traced the origin of code snippets and performed quantitative and qualitative analysis on code clone subjects.

B. Mining Stack Overflow and GitHub

Vasilescu et al. [24] were among the first researchers to explore the interaction between Stack Overflow and GitHub activities. The authors have linked Stack Overflow and GitHub user accounts to explore development activities on both platforms. Building on the work of Vasilescu et al. [24], Badashian et al. [25] investigated influence, involvement, and contribution across the same two platforms, by correlating activities on GitHub and Stack Overflow users. Lee and Lo [26] took Badashian et al. [25]'s work one step further and analyzed developer interests across the same two platforms, GitHub and Stack Overflow. Further, Vasilescu et al. [27] investigated how mailing lists and Stack Exchange provide knowledge sharing in open source software communities. Barua et al. [28] studied the process of sharing and gaining knowledge through being part of and interacting with like-minded developers in open source software communities.

Considering the popularity of GitHub and Stack Overflow in the software engineering research community, Gousios et al.

[29] have published the *GHTorrent project*, which is a public mirror dataset of all public projects available on GitHub. Many researchers have since leveraged the *GHTorrent* dataset. Some of the most interesting research efforts have been focused on the analysis of drive-by-commits, test incentives and pull-based development [30] [31] [32] [33] [34] [35].

In 2018, Baltes et al. [11] have created and released the *SOTorrent* dataset based on the official Stack Overflow data dump. *SOTorrent* stored version histories of Stack Overflow data on two separate levels: entire posts and individual code and text blocks. What contributed to the fast-growing popularity of *SOTorrent* is the ability to link Stack Overflow posts to GitHub repositories by detecting hyperlink references from GitHub files to Stack Overflow posts. Treude and Wagner [36] used *SOTorrent* and studied the characteristics of GitHub and Stack Overflow text corpora to predict good configurations for LDA models built on such corpora.

Many researchers have mined Stack Overflow for various purposes [37]–[49]. For example, Greco et al. [43] mined SO along with IDE tools to predict developer behaviour and developed a recommendation system for useful content on the platform. Zou et al. [44] use topic modeling to analyze non-functional requirements on SO and how they evolve. Like many previous efforts, we leverage the rich data from Stack Overflow and GitHub and study the change patterns of the snippets.

IV. Methodology

A. Datasets

To understand the evolution of snippets on both platforms, we mine two datasets, SOTorrent and GHTorrent.

SOTorrent [6] is an open dataset based on the official Stack Overflow data dump. SOTorrent provides access to the version history of Stack Overflow content at the level of a whole post and an individual post block [11]. A post in SOTorrent is defined as one entity in the discussion thread such as a question, comment, or answer to the question. Thus, a discussion thread on Stack Overflow is a set of posts in SOTorrent. Further, a post is divided into blocks. A post block can be one of the two types: 1) *code block* or 2) *text block*. A post block typically includes code snippets (code blocks) and the textual content (text blocks) and is dependent on the author's formatting style of the post. Our definition of a SO snippet may refer to a code block, a text block, or both.

The version history of a Stack Overflow post in SOTorrent is defined in terms of the version history of blocks. Each edit, whether an existing post block is edited, created or deleted, creates a new version to the entire post and that particular post block. Along with the version history of content, SOTorrent links SO posts to external resources in two ways: 1) by extracting external URLs from post content, and (2) by providing a table with links to SO posts found in the source code of GitHub projects, called `PostReferenceGH` in SOTorrent's schema. This linkage of SO posts to GitHub projects is critical for our research as it allows us to map SO posts to GitHub projects and identify all SO snippets (code and text blocks) that were reused in GitHub projects. Each code block that is introduced with a reference to SO post to a GitHub project is referred as a reused code snippet.

GHTorrent [31] is an online queryable mirror of the GitHub's events. These events include meta-information about various interactions happening on the platform, for example, commit to a project, the opening of a pull/issue request. While GHTorrent is suitable for characterizing the projects that are reusing Stack Overflow snippets, it is not sufficient for further in-depth analysis for the following reason. GHTorrent consists of the events from GitHub and metadata related to those events (e.g., commit), but code changes (as code `diffs`) related to those events are not present in the dataset. For the analysis of the code snippet change history within the GitHub space, events related to code snippets need to be identified using code `diff`. Therefore, our main effort is related to extracting the history of changes from the commit history of the code snippets that are reused from Stack Overflow discussions.

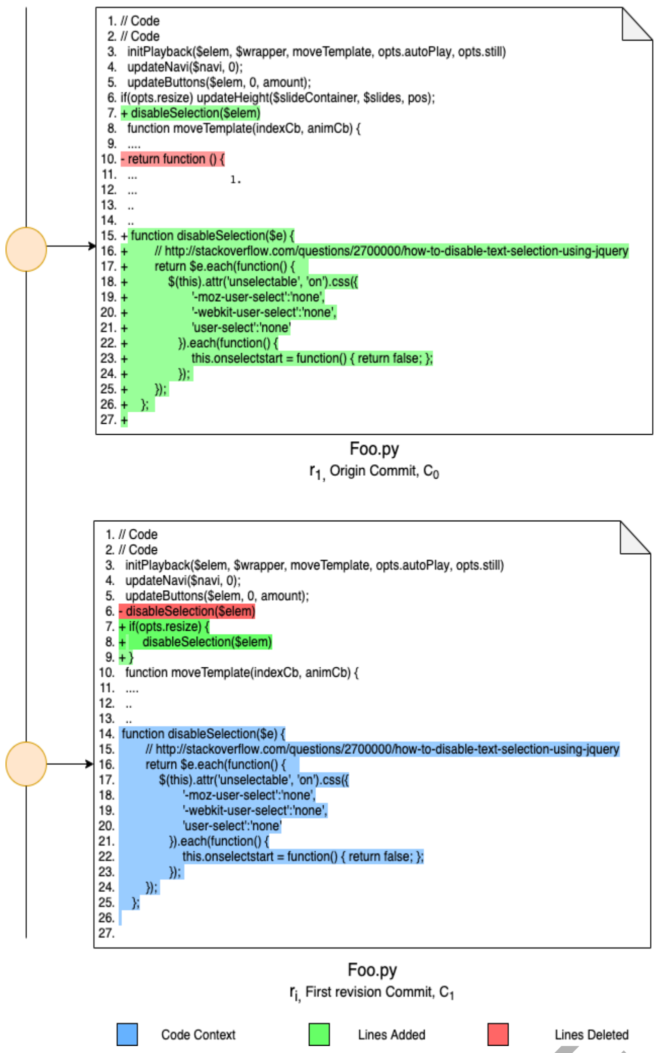
To extract the change history of code snippets in GitHub, we mine GitHub projects and create a new dataset called *GHCodeSnippetHistory*. The majority of the previous research work on mining SO and GH data focused on specific sample of the projects (e.g., Java, Python). Our aim is to collect a sample dataset that is representative of the entire population of the GitHub projects that refer to SO posts. We now describe the approach of collecting a representative sample of the GitHub projects and how the change history of code snippets was extracted.

B. Creating Dataset

To understand how the change history of code snippets is extracted from the GitHub projects, we need to introduce two main components: the concept of *Code Context* and the *Code Miner* tool.

1) Code Context: Since we focus on the change history of code, the unit of the analysis is commit. When we examine the code that contains a reference to a Stack Overflow post, we first need to identify the original commit which has introduced that reference. We call it the *origin commit*. Since GitHub stores the version history of all files, we can determine all the commits that have touched any given file. Given we know all the commits for a file and the changes they introduced, in chronological order, we can identify the *origin commit*. The earliest commit that has introduced SO reference is labelled as the *origin commit* C_0. The changes introduced in the origin commit become the code snippet of interest, and we would like to understand how this code evolves overtime. Again, this code snippet is not claimed to be a clone of the corresponding SO snippet but is assumed to be the reused (i.e., adapted) code snippet.

Suppose that a changeset δ_0 that is a part of the *origin commit* C_0, touches file *f*, that has a reference to SO post, and transforms its revision r_1 to r_2. First, we detect lines $L \in \delta_0 = \delta_{a0}$ that have been added by δ_0 in r_2. These lines attribute to the first version of the reused code snippet. To locate them, we use *git show origin-commit-hash* command.



```
// Code
// Code
  initPlayback($elem, $wrapper, moveTemplate, opts.autoPlay, opts.still)
  updateNavi($navi, 0);
  updateButtons($elem, 0, amount);
if(opts.resize) updateHeight($slideContainer, $slides, pos);
+ disableSelection($elem)
  function moveTemplate(indexCb, animCb) {
  ....
- return function () {
  ...
  ...
  ..
  ..
+ function disableSelection($e) {
+       // http://stackoverflow.com/questions/2700000/how-to-disable-text-selection-using-jquery
+       return $e.each(function() {
+           $(this).attr('unselectable', 'on').css({
+                   '-moz-user-select':'none',
+                   '-webkit-user-select':'none',
+                   'user-select':'none'
+               }).each(function() {
+                   this.onselectstart = function() { return false; };
+               });
+       });
+   };
+
```

```
// Code
// Code
  initPlayback($elem, $wrapper, moveTemplate, opts.autoPlay, opts.still)
  updateNavi($navi, 0);
  updateButtons($elem, 0, amount);
- disableSelection($elem)
+ if(opts.resize) {
+     disableSelection($elem)
+ }
  function moveTemplate(indexCb, animCb) {
  ....
  ..
  ..
 function disableSelection($e) {
       // http://stackoverflow.com/questions/2700000/how-to-disable-text-selection-using-jquery
       return $e.each(function() {
           $(this).attr('unselectable', 'on').css({
                   '-moz-user-select':'none',
                   '-webkit-user-select':'none',
                   'user-select':'none'
               }).each(function() {
                   this.onselectstart = function() { return false; };
               });
       });
   };

```

Foo.py
$r_{1,}$ Origin Commit, C_0
Foo.py
$r_{i,}$ First revision Commit, C_1
🟦 Code Context
🟩 Lines Added
🟥 Lines Deleted

Fig. 1: Extracting code context from commits.

This subset δ_{a0}, defines a *code context* (Δ), i.e., a set of lines added to the origin commit $\Delta = \delta_{a0}$. From this point on, we want to identify all commits that change the code lines of this code context.

Now consider a revision r_j to *f*. Let C_j be the commit that upgrades file from r_{j-1} to r_j. Let δ_j be the changeset of C_j, and $\delta_{dj} \subseteq \delta_j$ is set of deleted lines, and δ_{aj} be added lines in C_j. If, a code line, $L \in \delta_{dj}$, i.e., the line is a part of the deleted lines, and $L \in \Delta$, i.e., the line is a part of the code context as well, then C_j is modifying the current context and is added to the dataset, otherwise, revision r_j is not of interest to us as it is not modifying the reused snippet. If C modifies the context, we update the context as $\Delta = (\Delta \setminus \delta_d) \cup \delta_a^{'}$, where $\delta_a^{'} \subseteq \delta_a$, and is collection of all the lines in δ_a, that are in close proximity to the already present lines in Δ, i.e., the code context.

Figure 1 demonstrates this concept via an example from a GitHub project (file is renamed for the sake of simplicity). Once we extract all commits from the file history, we identify C_0 by looking for the commit whose changeset has introduced a reference to SO post, i.e., the following URL: http://stackoverflow.com/questions/27000/how-to-disable-text-selection-using-jquery.

All lines added in this origin commit are shown in green in Figure 1 and form *code context* for next revision. Then, we identify another commit C_1 that happens after C_0 but consists of the changeset, which touches some of the lines added to the origin commit and is present in the code context. This commit is considered to be a part of the code snippet's change history. The code context is now updated by removing deleted lines and adding lines that were introduced next to the existing lines (i.e., code context consists of green and blue lines for next revision). We continue to walk through all the commits of the file in its commit-graph. From these commits, we select a subset of commits that modify the code context, thus, capturing the change history of a code snippet.

2) Code Miner: Code Miner is a tool developed for creating the *GHCodeSnippetHistory* dataset. Code Miner operates on seed data points that are a subset or the entire table `PostReferenceGH` from SOTorrent. Essentially, Code Miner clones a git project and then mines its file history using the concept of code context.

The URLs to GitHub projects that reference SO posts are stored in `PostReferenceGH` table and look similar to the following URL: https://raw.githubusercontent.com/RationalAsh/freeIMU/master/debug/decode_float.py. To derive the project information from a URL, it can be split into several elements as shown in Table III. For a given project, i.e., USER/REPO-NAME, Code Miner aggregates all SO references to avoid multiple project clones. Code Miner then sends a GitHub's REST API request to gather general information on the project such as the repository size, number of files, clone URL. If the request is successful, this means that the project is public and can be cloned without infringement of copyright. The URL of such a request has the following signature: https://api.github.com/repos/USER/REPO-NAME.

TABLE III: Deriving GitHub project information.

Element	Example
HEADER	*https://raw.githubusercontent.com*
USER/REPO-NAME	*RationalAsh/freeIMU/*
BRANCH	*master*
FILE-PATH	*debug/decode_float.py*

After cloning the repository, Code Miner locates the file with the reused code snippet in the codebase. This is the file that has a URL to Stack Overflow post.

Code Miner runs `git blame` on the identified file and creates a `blame` file. `Git blame` produces the entire commit-graph of a given source file. This commit-graph is then sorted in chronological order. Using the concept of code context, a subset of commits from the commit-graph is selected since these commits contribute to the change history of the reused code snippet. Code Miner extracts the entire metadata of the commit along with the changes in the file of interest (i.e., changes to the reused snippet).

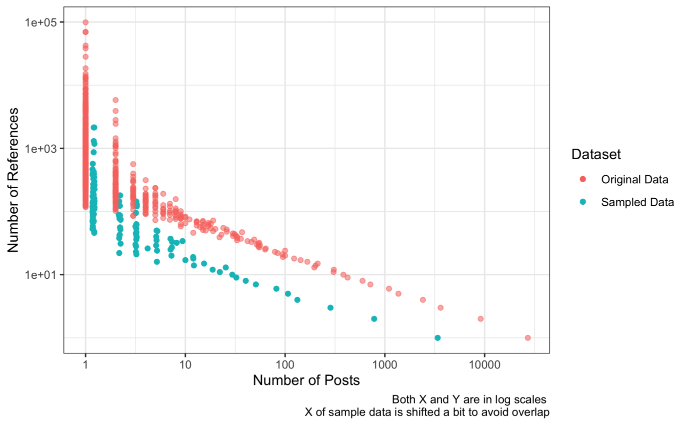


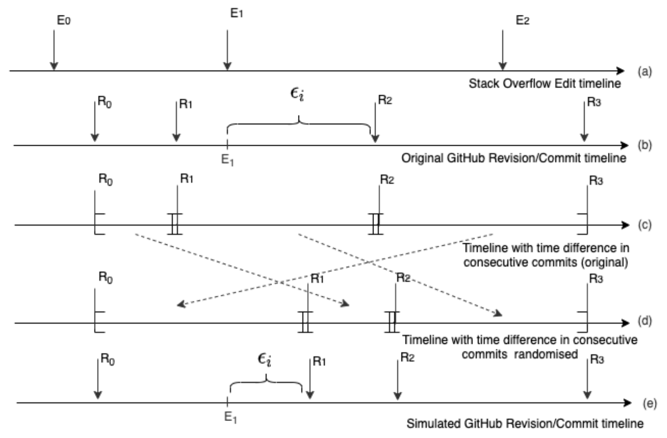
Fig. 2: Distribution of a sampled dataset.

3) Project Sampling: In SOTorrent (version 2019/06/21), `PostReferenceGH` table has a catalogue of 6.5 million references used in public GitHub projects that refer to 150,000 Stack Overflow posts. These references span over 440,000 GitHub projects with 1,678 different file formats identified from the file extensions. Mining 440,000 repositories and creating a change history of all reused snippets is not feasible due to time, computing, and storage constraints. Yet, we want to preserve the generalization of our results derived from the reduced (i.e., sampled) GitHub data, as well as reduce sampling bias. Our sampling technique combines a stratified random sampling [50] and the technique used by Treude et al. [51]. We first sample SO posts and then the GitHub projects in which these SO posts are attributed [52]. To sample SO posts, we follow Treude et al. [51]. After sampling all posts (i.e., postIDs), we applied a stratified random sampling [50] to sample corresponding GitHub projects in which these posts are referenced.

To be able to scale our approach, we set the desired dataset size to 40,000 SO references. Figure 2 illustrates that our sampled data is representative of the distribution of SO references in GitHub. Our final dataset, called *GHCodeSnippetHistory* dataset contains 22,900 projects, 33,765 SO references mapped to 4,634 SO posts, and total of 73,322 commits. The number of projects is smaller than the original target in the sampling algorithm, i.e., 40,000. The reason for this discrepancy is that either the sampled project or the file containing a SO link was deleted.

C. Time Series Analysis

Our RQ1 and RQ2 represent the macroscopic view of code evolution and edit/revision activities on Stack Overflow and GitHub. The microscopic view takes a temporal aspect into account by considering code revisions on GitHub through committing and, on Stack Overflow, through edit activities, as time series. These microscopic views can answer the dependency of code changes on both platforms. Thus, to study the interaction between change activities on both platforms, we follow the approach of Xuan et al. [53]. According to their approach, using the original time series of revision activities



E0
E1
E2
(a)
Stack Overflow Edit timeline
R0
R1
ϵi
R2
R3
E1
(b)
Original GitHub Revision/Commit timeline
R0
R1
R2
R3
(c)
Timeline with time difference in consecutive commits (original)
R0
R1
R2
R3
(d)
Timeline with time difference in consecutive commits randomised
R0
ϵi
R1
R2
R3
E1
(e)
Simulated GitHub Revision/Commit timeline

Fig. 3: Generating a simulated time-series of GitHub commit activities.

as the control group, several simulated activities are produced, and then simulated series are compared with the original control group, and the statistical difference is determined.

Consider the timeline of Stack Overflow edit activities of a particular pair of a post and the timeline of its reused code snippet on GitHub as shown in Figure 3 (a) and (b). Let $\mathcal{A}$ and $\mathcal{B}$ be two activities we would like to compare (e.g., E for edit of Stack Overflow post and R for revision of the reused code snippet). For every event E_i of $\mathcal{A}$ we measure the impact latency ϵ_i as the difference between the earliest event of $R \in \mathcal{B}$ following E_i (see Figure 3 (a)). The sequence ϵ characterises the relationship between $\mathcal{A}$ and $\mathcal{B}$. Next, to study whether the sequence of $\mathcal{B}$ events, i.e., revisions on GitHub (R_i), for a particular edit on Stack Overflow E_i could have occurred by chance, we create m random permutations of $\mathcal{B}$ events $(\mathcal{B}_1, \ldots, \mathcal{B}_m)$. While reshuffling, we make sure that the time difference or "idling period" between two consecutive revisions (R_i, R_{i+1}) of the reused code snippets remains the same, while the order of the "idling periods" is randomised (as shown in Figure 3 (d)). Let $\epsilon_1, \ldots, \epsilon_m$ be series of impact latencies corresponding to $\mathcal{B}_1, \ldots, \mathcal{B}_m$ (e.g., as shown in Figure 3 (e)).

Finally, we aggregate all the impact sequences such as ϵ for all such pairs in the dataset into $E^{\mathcal{B}}$. Then, we compare $E_{\mathcal{B}}$ with each one of $E^{\mathcal{B}_1}, E^{\mathcal{B}_2}, ..E^{\mathcal{B}_m}$, where $E_{\mathcal{B}}$ is referred as *control sample* while others, $E^{\mathcal{B}_i}, i \in [1, 10]$ as *simulated sample*. Finally we compare each simulated sample with control sample using t-tests. The t-test [54] is a parametric statistic that helps to determine how significant is the difference between two groups. The results should be as follows:

1) If $\mathcal{A}$ and $\mathcal{B}$ are independent from each other, $E^{\mathcal{B}}$ will be statistically indistinguishable from its simulated counterparts.
2) If $\mathcal{A}$ and $\mathcal{B}$ are correlated in some manner, $E^{\mathcal{B}}$ will be statistically longer or shorter than its simulated counterparts, indicating acceleration and de-acceleration.

We apply this approach for answering our RQ4 related to the dependence of the GitHub revisions on the Stack Overflow post edits.

V. Results

We now present the results of our analysis and provide answers to the research questions.

A. RQ1: To what extent do snippets evolve on GitHub and Stack Overflow?

To investigate to what extent snippets evolve, we concentrated on the number of changes per snippet on both platforms, i.e., Stack Overflow and GitHub. Each change in a reused code snippet in GitHub is referred to as a **revision** or as an **edit** for Stack Overflow snippets. Table IV reports the number of revisions for GH code snippets and number of edits for SO snippets. By looking at the column "All GH snippets", we can observe that the majority of reused code snippets in GitHub do not get revised over time. We decided to separate GitHub code snippets into two categories:

1) *Unchanged GH code snippets*: Code snippets that are never updated after being introduced to a GH project.
2) *Changed GH code snippets*: Code snippets that are revised at least once after their inception.

Unchanged code snippets are code snippets that have only one commit in the project's history, the commit in which they were introduced to the project. We found that these unchanged code snippets represent 77.98% (26,330 in total) of the total reused code snippets. On the contrary, on Stack Overflow, only 23.7% of snippets undergo no edits after their creation.

Figure 4 (left) shows the distribution of *changed* GitHub code snippets (with 5% of outliers being removed), while (right) shows the distribution of edits in Stack Overflow snippets. Combining these distribution trends with the descriptive statistics in Table IV, we can say that SO snippets undergo slightly fewer edits than the adapted code snippets that do change in GitHub (mean values are 4.61 and 6.54 for SO and GH, respectively).

Answer to RQ1: *22% of the reused code snippets evolve in GitHub, while 76.3% of snippets evolve on Stack Overflow. On average, Stack Overflow snippets undergo fewer changes compared to their evolving counter parts on GitHub.*

B. RQ2: How and how often do snippets evolve in GitHub and Stack Overflow?

To study how snippets evolve, we consider the change type and size of each SO and reused GH snippets' revision/edit. Table V and Table VI report summaries of multiple revisions for Stack Overflow snippets and GitHub code snippets respectively, showing the median time difference between

TABLE IV: Changes in GH code snippets and SO snippets.

Metric	All GH snippets	Changed GH snippets	SO snippets
Median	1.00	3.00	3.00
Mean	2.20	6.54	4.61

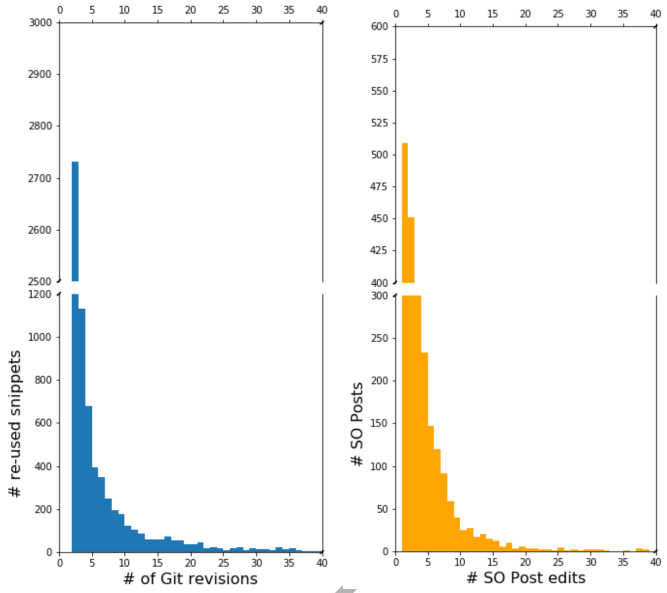


Fig. 4: Revisions in changed code snippets on GitHub (left) and edits in Stack Overflow snippets (right).

revisions and the change size. In these tables, snippets are categorized by their edit/revision number as represented by the column "Edt#" or "Rev.#" for SO and GH, respectively. The 0^{th} edit/revision category means the creation/introduction of the snippet on the corresponding platform. Any further changes after 7^{th} revision are merged in 7+ category. We also report the change type that is specific to each platform. For GitHub, the change type is either ADD, DELETE, MODIFY, or RENAME depending on whether the code lines were added, deleted or modified, or file being renamed for the reused code snippet in the corresponding revision. Similarly, for Stack Overflow, the change type is either TEXT or CODE, indicating whether the text or code block of the post was revised during the edit. Column *"Major Change Type"* indicates the change type of the majority of snippets for a particular revision, while *"Second Major Change Type"* denotes the runner up change type for all snippets. From Table V, we can observe that the majority of the edits to Stack Overflow content have the change size of around 2 lines for the text block and around 1 line for the code block. The major change type for SO snippets is TEXT indicating that the description of the post is more prone to changes. Table VI shows that the vast majority of the reused code snippets are of MODIFIED type over multiple revision, with the change size of 24 LOCs, on average.

To investigate how often snippets are changed on each platform, we look at the time difference between two consecutive revisions/edits. To show the timeline range over which the consecutive revisions can happen, we report the minimum (in hours, H) and maximum time (in years, Y) difference along with the median time (in days, D) in Table V and Table VI. We observe that the time interval is shorter between revisions of the reused code snippets (median time is between 4.3 and 11 days), while the time interval is longer for the SO snippet

TABLE V: Edit characteristics for Stack Overflow snippets.

Edt. #	Major Change Type	2nd Major Change Type	Time (in H,D,Y) Min	Max	Mdn	Change Size Text	Code
0	TEXT&CODE	TEXT	–	–	–	3	8
1	TEXT	TEXT&CODE	< 1H	9.8Y	3.5D	2	2
2	TEXT	CODE	< 1H	10Y	143D	2	1
3	TEXT	TEXT&CODE	< 1H	9.1Y	149D	2	1
4	TEXT	CODE	< 1H	10.3Y	111D	2	1
5	TEXT	TEXT&CODE	< 1H	8.3Y	144D	2	1
6	TEXT	CODE	< 1H	7.3Y	110D	2	1
7+	TEXT	CODE	< 1H	6.6Y	6.2D	1	2

TABLE VI: Revision characteristics for GitHub code snippets.

Rev. #	Major Change Type	2nd Major Change Type	Time (in H,D,Y) Min	Max	Mdn	Change Size [LOC]
0	ADD	MODIFY	–	–	–	930
1	MODIFY	DELETE	< 1H	5Y	11D	30
2	MODIFY	RENAME	< 1H	4.7Y	10D	21
3	MODIFY	DELETE	< 1H	5.8Y	7.8D	23
4	MODIFY	DELETE	< 1H	5.6Y	6.7D	23
5	MODIFY	DELETE	< 1H	4.3Y	6D	23
6	MODIFY	RENAME	< 1H	3.7Y	6.9D	26
7+	MODIFY	DELETE	< 1H	4.9Y	4.3D	22

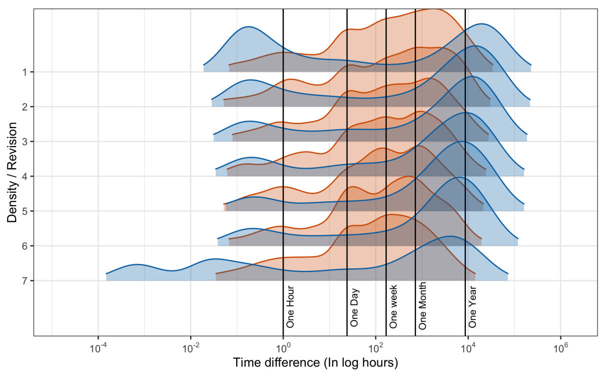


Fig. 5: Density plot of revision distributions over time on GH (in blue) and SO (in orange).

edits (median time is between 3.5 and 149 days). Also, the results from these tables suggest that the evolution timeline of Stack Overflow snippets is longer compared to the timeline of the GitHub code snippets.

Figure 5 shows the distribution of time difference between subsequent revisions on both platforms, for all snippets. These distributions are again classified by their revision numbers. For example, in Figure 5, 2^{nd} revision, represented by number 2 on the y-axis and its corresponding density curve, shows the distribution of the time difference between second and first revisions for all snippets. As shown in Figure 5, the revision/edit time difference distribution follows a unique pattern for each platform across multiple revisions, i.e., the distribution curves representing each of the seven revisions on GitHub (in blue) are of a similar pattern. However, these curve patterns are different for each platforms. We observe that GitHub distribution curves resemble bimodal distributions, however, further analysis is needed to confirm this.

Answer to RQ2: *On average, Stack Overflow snippets undergo minor changes of 2 lines to the textual description and 1 line to the code block of the answer, while the change size of the GitHub code snippets is 24 LOCs. The major change type is TEXT and MODIFIED for Stack Overflow snippets and reused code snippets, respectively. Overall, the evolution timeline of the SO snippets is longer compared to the timeline of the reused code snippets.*

C. RQ3: Who is driving the evolution of snippets on both platforms?

GitHub and Stack Overflow are collaborative platforms, i.e., a lot of social interaction and discussion happens around code and software development. One of the dominant trends, in particular on Stack Overflow, is that posts and code snippets can become the product of collective ownership as multiple authors change the content over time. To understand who is contributing to the content changes, i.e., posts on Stack Overflow and the code snippets that are reused on GitHub, we classify authors into original vs. non-original. *Original authors* are the users who created the content in the first place, e.g., created a post on Stack Overflow or reused a SO snippet in a GitHub project. In contrast, *non-original author* is anyone but the original author.

Figure 6 (left) shows the percentage of SO snippet edits by original vs. non-original author, while Figure 6 (right) shows the portion of commits on the reused code snippets for a particular revision by original vs. non-original author. We notice that the original authors (in blue) take up most of the ownership of the content as they remain the primary drivers of the content (post or code) maintenance on both Stack Overflow and GitHub platforms.

Answer to RQ3: *The author of the original snippet remains the leading contributor to its maintenance and evolution on both platforms.*

D. RQ4: To what extent do Stack Overflow snippets and the reused code snippets co-change?

While our answers to previous questions provide evidence that snippets evolve on both platforms, next, we wanted to explore to what extent the revisions and edits to these snippets might be happening in parallel, i.e., co-change. Figure 7 shows the co-change of snippets on both platforms, as a function of time. The x-axis timeline of revisions/edits represents the number of days since 2008-08-05 till 2019-09-01. With time, the amount of posts created on Stack Overflow is growing, similarly the number of snippets that are reused from SO and integrated to GitHub projects is increasing. Further, we observe a time lag between the Stack Overflow curve and the GitHub curve representing the delay between the creation of SO snippets and their reuse in GitHub. Overall, it takes a while (at least 1,000 days) for snippets to be reused in GitHub projects. In time, SO content becomes more stable as developers start reusing it in their GitHub projects. We also notice that the majority of the reused code snippets do not change. Eventually, the number of posts created on SO becomes stable (i.e., the posts in our dataset), and their content also gets more stable as GitHub projects start reusing the content from SO. We can observe, yet again, that the majority of adapted code snippets do not evolve.

Looking at the change history of all reused code snippets and their SO counterparts as time-series data, as shown in

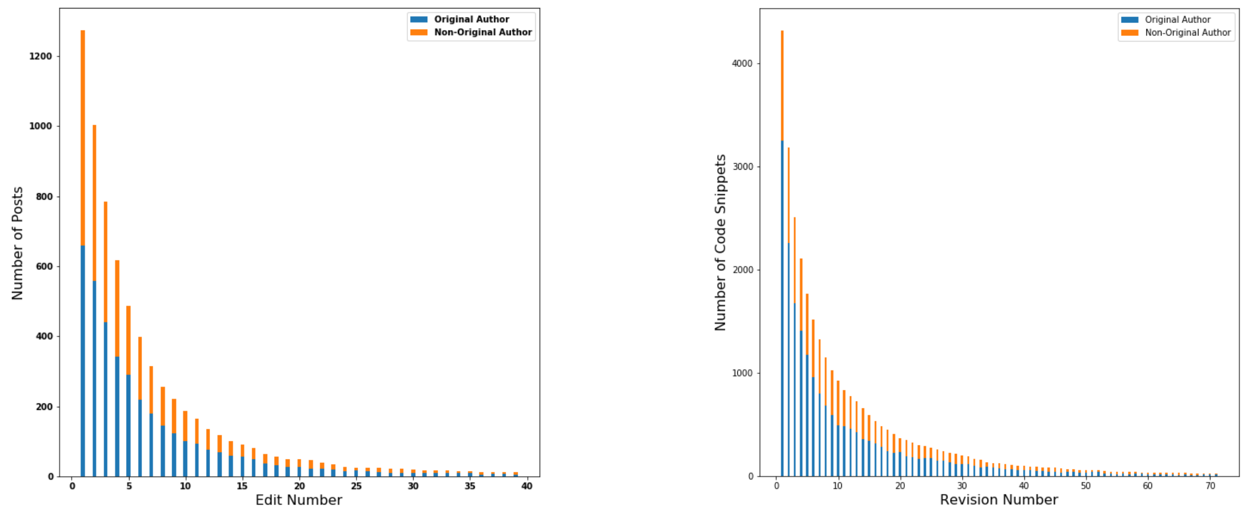


Fig. 6: Contributors on Stack Overflow (left) and GitHub (right).

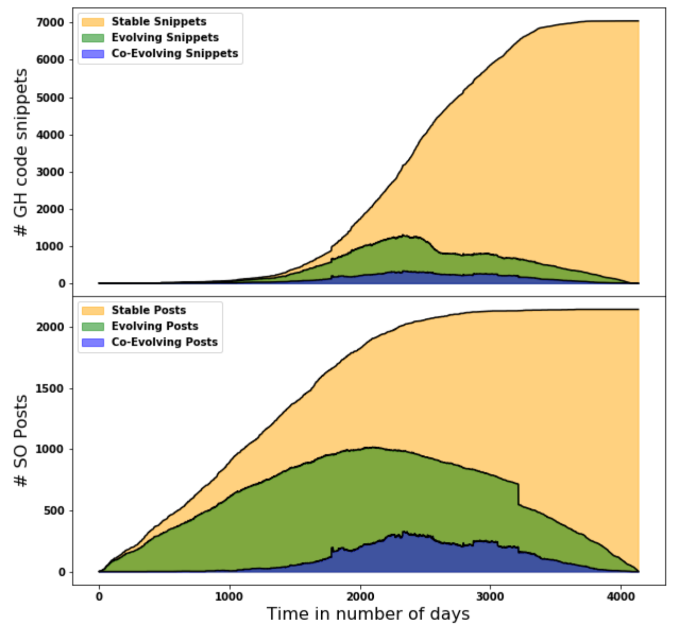


Fig. 7: Co-change timelines of SO snippets and reused code snippets.

Figure 7, we observe different peaks at different timestamps. For GitHub, the peak of the evolving snippets (green curve) is at 41.2% of the height of the stable snippets (yellow) curve at that time, denoting that at most 41.2% of the reused code snippets were changed at any given time. Similarly, for Stack Overflow, the peak of the evolving snippets (green curve) is at 53.3% of the height of the yellow curve, implying that only about half of the referred snippets were evolving at that any given time. Finally, blue curves on the graphs of both platforms indicate the percentage of snippet pairs that co-change at a particular timestamp. Since the scales of the two graphs are different, the height of the peaks seems different. However, the blue curve is the same on both graphs. The highest peak of the blue curve is at 10% of the reused snippets. This means that at most 10% of the reused code snippets and their SO counterparts were evolving together across two platforms at any moment of time.

Based on these results, we can say that the reused code snippets evolve while their counterpart posts evolve on Stack Overflow because their timelines overlap. But it is not clear whether the changes to Stack Overflow content are triggering revisions of the reused code snippets. To determine whether there is a dependency between Stack Overflow evolution and GitHub evolution, i.e., there is a relation between the edit activities on Stack Overflow and the commit activities on GitHub, we follow the approach proposed by Xuan et al. [53] and Vasilescu et al. [24] as described in Section IV-C. In this approach, we compute impact latencies ϵ for each pair of SO post and reused code snippet. This latency denotes the time difference between edit activity on Stack Overflow and revision activity on GitHub.

Further, following Vasilescu et al. [24], we split computed impact latencies ϵ_i's of each pair into quarters by the number of GitHub revisions. Our t-test results are presented in Table VII for two simulated samples. Based on these results, we conclude that the reused code snippets in GitHub evolve independently of their corresponding Stack Overflow snippets. This means that Stack Overflow edits do not impact the changes of the reused code snippets in GitHub and vice versa.

Answer to RQ4: *While having overlapping evolution timelines, the reused code snippets and their corresponding SO snippets evolve independently of one another.*

TABLE VII: T-test results for GitHub revisions and Stack Overflow edits.

Sample	**Q1**		**Q2**		**Q3**		**Q4**	
	μ	p	μ	p	μ	p	μ	p
Control	779.94	–	602.01	–	490.40	–	379.43	–
Simulated 1	779.94	0.99	602.09	0.99	496.67	0.71	381.34	0.88
Simulated 2	779.94	0.99	600.98	0.96	501.13	0.54	379.80	0.97

VI. Discussion

A. Findings

The goal of this work was to better understand how online snippets evolve in Stack Overflow compared to how the reused code snippets evolve in GitHub. We tried to identify the evolution patterns on each platform and determine whether co-change of snippets happens across two platforms. We found that snippets evolve on both platforms, i.e., GitHub and Stack Overflow, simultaneously and independently. The pace of evolution is different on both platforms, while initial edits and revisions on both platforms happen quite fast.

While we found that the evolution of snippets happens on both platforms, we have not investigated why snippets undergo multiple corrections either on Stack Overflow or GitHub. Future research should focus on studying the factors that drive these changes.

B. Implications

Maturity of SO content. Combining the evolution history and the usage history of a Stack Overflow discussion, predictive models can be developed for assessing the quality of the SO content. Adding "maturity" score to posts in Stack Overflow can help developers determine whether code is ready (or not) to be used and adapted to their own projects.

A dataset for code clone detection. We mined GitHub projects to create a code history that stores code diffs along with the related metadata. Code, collected as a part of data mining, is mostly unexplored, yet it provides ready-to-be-used dataset for clone detection across a large number of projects.

Tool development for tracking SO changes. Our results suggest that evolution on Stack Overflow happens at a slower pace compared to GitHub. Manual tracking of SO changes is infeasible. Therefore, we recommend the development of automated solutions to help developers keep track of the snippets they use. For example, an application to be integrated with CI (e.g., CircleCI) or code coverage (e.g., Sonar) tools can query the SOTorrent-like database to check for changes in the SO snippets that are used in the codebase. If a change is found, a warning can be issued for a pull request to keep the author informed of the change. Such tools can alleviate the problem of bugs migrating from SO to GH projects.

C. Threats to Validity

Our study is subject to limitations and threats to validity.

Limitations of Code Miner. To recreate a change history of the reused code snippets, Code Miner relies on the `git history` of the project. Often the commit history is modified via `git --rebase` option. In such scenarios, it is challenging to recreate original commit history. Such altered history significantly reduces the number of commits per code snippet while increasing the number of changed lines in the merged commit. Also, if code from third party libraries is borrowed in the project, and the SO reference is present in such code, Code Miner is not able to recreate the history of such code.

Sampling. While our sampling technique aims at reducing sampling and selection bias, we had to exclude one project named *cdnjs* from our dataset due to its size (100GB) and technical issues (disk space and computational resources) we faced during the project cloning step.

SO snippet definition. Our definition of the SO snippet is rather broad and refers to answers that may include code examples, pseudo code, design, algorithms, or be text-only. "Origin commit" defines whether code or information was adapted from SO by including an attribution [52] (i.e., URL to a SO post) to source code. We do not check whether the code was copied or not. By applying code clone detection techniques one would be able to provide better insights on *how* snippets are reused.

SO reference mapping. Our analysis is based on the comparison of each GitHub code snippet with its corresponding SO snippet. If a SO reference points to a SO question post, we have mapped the reference to the accepted answer or highly voted answer (if no accepted answer is available). According to Stack Overflow, the accepted answer provides the best solution.

Code ownership. When analyzing revisions of the reused code snippets, we found some revisions not having same "Committer ID" and "Author ID". The author is the person who wrote the code, while the committer is the person who committed code on behalf of the author [55]. In this work, while identifying contributors, we focused on the commit authors only, excluding the committers from the analysis.

Lack of qualitative analysis. GH snippets are much larger than original SO snippets; we observed that initial commits may include a merge of a large number of commits or even history rebases. In-depth qualitative studies of commits together with code clone detection/analysis would be needed to better understand how SO content is reused by developers. The results on GitHub code snippet evolution should be interpreted as a higher-bound, i.e., the real "reused" code is likely to be smaller indicating that the GitHub code evolution can be even less frequent than the results presented in the paper.

VII. Conclusion

In this work, we investigated the change histories of snippets on Stack Overflow and GitHub. We studied SO snippets that have been reused in GitHub projects by mapping data sources, SOTorrent and GHTorrent, and building a new dataset that provides a mapping of these data to the revision history of the reused code snippets along with code diffs. Our results demonstrate that 76.3% of snippets evolve on Stack Overflow, while only 22% of the reused code snippets evolve in GitHub. However, Stack Overflow snippets undergo fewer and minor changes compared to their evolving counterparts on GitHub. We analyzed the ownership of the content on both platforms and found that the creator of the content plays a critical role in the evolution of snippets. When exploring the possibility of code co-change, we found that the percentage of co-changing snippets is small on the overall timeline. Moreover, snippets evolve independently on each platforms (e.g., Stack Overflow edits have no impact on GitHub revisions).

REFERENCES

[1] Stack Overflow, "Stack Overflow," https://stackoverflow.com/, 2020, [Online; accessed 05-January-2020].

[2] ——, "Stack Overflow Insights," https://insights.stackoverflow.com/, 2019-12, [Online; accessed 05-January-2020].

[3] S. M. Nasehi, J. Sillito, F. Maurer, and C. Burns, "What makes a good code example?: A study of programming QA in Stack Overflow," in *Proc. of the International Conference on Software Maintenance*, 2012, pp. 25–34.

[4] GitHub, "GitHub," https://github.com/, 2020, [Online; accessed 05-January-2020].

[5] Stack Exchange, "A New Code License: The MIT, this time with Attribution Required," https://meta.stackexchange.com/questions/272956/a-new-code-license-the-mit-this-time-with-attribution-required?cb=1, 2020, [Online; accessed 05-January-2020].

[6] S. Baltes, C. Treude, and S. Diehl, "SOTorrent: Studying the origin, evolution, and usage of stack overflow code snippets," in *Proc. of the International Conference on Mining Software Repositories*, 2019, pp. 191–194.

[7] C. Ragkhitwetsagul, J. Krinke, M. Paixao, G. Bianco, and R. Oliveto, "Toxic code snippets on stack overflow," *IEEE Transactions on Software Engineering*, pp. 1–1, 2019.

[8] L. An, O. Mlouki, F. Khomh, and G. Antoniol, "Stack Overflow: A code laundering platform?" in *International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 2017, pp. 283–293.

[9] F. Fischer, K. Böttinger, H. Xiao, C. Stransky, Y. Acar, M. Backes, and S. Fahl, "Stack Overflow Considered Harmful? The Impact of Copy Paste on Android Application Security," in *IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 121–136.

[10] M. Verdi, A. Sami, J. Akhondali, F. Khomh, G. Uddin, and A. Karami Motlagh, "An Empirical Study of C++ Vulnerabilities in Crowd-Sourced Code Examples," *IEEE Transactions on Software Engineering*, pp. 1–1, 2020.

[11] S. Baltes, L. Dumani, C. Treude, and S. Diehl, "Sotorrent: Reconstructing and analyzing the evolution of stack overflow posts," in *Proceedings of the 15th International Conference on Mining Software Repositories*. ACM, 2018, pp. 319–330.

[12] Sebastian Baltes, "SOTorrent Dataset," https://empirical-software.engineering/projects/sotorrent/, 2019-12, [Online; accessed 05-January-2020].

[13] H. Gonzalez, N. Stakhanova, and A. A. Ghorbani, "Measuring code reuse in android apps," in *Proc. of the Annual Conference on Privacy, Security and Trust*, 2016, pp. 187–195.

[14] A. Lotter, S. A. Licorish, B. T. R. Savarimuthu, and S. Meldrum, "Code Reuse in Stack Overflow and Popular Open Source Java Projects," in *Proc. of the Australasian Software Engineering Conference*, 2018, pp. 141–150.

[15] Z. Wang, M. Zhou, and H. Mei, "Towards an Empirical Reuse Approach for the Software Evolution: A Case Study," in *Proc. of the International Conference on Quality Software*, 2010, pp. 282–287.

[16] S. Baltes, R. Kiefer, and S. Diehl, "Attribution Required: Stack Overflow Code Snippets in GitHub Projects," in *International Conference on Software Engineering Companion (ICSE-C)*, May 2017, pp. 161–163.

[17] D. Yang, P. Martins, V. Saini, and C. Lopes, "Stack Overflow in GitHub: Any Snippets There?" in *Proc. of the International Conference on Mining Software Repositories*, 2017, pp. 280–290.

[18] T. Kamiya, S. Kusumoto, and K. Inoue, "CCFinder: a multilinguistic token-based code clone detection system for large scale source code," *IEEE Transactions on Software Engineering*, vol. 28, no. 7, pp. 654–670, 2002.

[19] J. Krinke, "A Study of Consistent and Inconsistent Changes to Code Clones," in *Proc. of the Working Conference on Reverse Engineering*, 2007, pp. 170–178.

[20] R. Abdalkareem, E. Shihab, and J. Rilling, "On Code Reuse from Stack Overflow," *Inf. Softw. Technol.*, vol. 88, no. C, pp. 148–158, Aug. 2017.

[21] D. Yang, P. Martins, V. Saini, and C. Lopes, "Stack Overflow in GitHub: Any Snippets There?" in *Proc. of the International Conference on Mining Software Repositories*, 2017, pp. 280–290.

[22] V. Saini, H. Sajnani, and C. Lopes, "Comparing Quality Metrics for Cloned and Non Cloned Java Methods: A Large Scale Empirical Study," in *Proc. of the International Conference on Software Maintenance and Evolution (ICSME)*, 2016, pp. 256–266.

[23] C. Kapser and M. W. Godfrey, ""Cloning Considered Harmful" Considered Harmful," in *Proc. of the Working Conference on Reverse Engineering*, 2006, pp. 19–28.

[24] B. Vasilescu, V. Filkov, and A. Serebrenik, "StackOverflow and GitHub: Associations between Software Development and Crowdsourced Knowledge," in *Proc. of the International Conference on Social Computing*, 2013, pp. 188–195.

[25] A. S. Badashian, A. Esteki, A. Gholipour, A. Hindle, and E. Stroulia, "Involvement, contribution and influence in github and stack overflow," in *Proceedings of 24th Annual International Conference on Computer Science and Software Engineering*. IBM Corp., 2014, pp. 19–33.

[26] R. K.-W. Lee and D. Lo, "Github and stack overflow: Analyzing developer interests across multiple social collaborative platforms," in *International Conference on Social Informatics*. Springer, 2017, pp. 245–256.

[27] B. Vasilescu, A. Serebrenik, P. Devanbu, and V. Filkov, "How social q&a sites are changing knowledge sharing in open source software communities," in *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing*. ACM, 2014, pp. 342–354.

[28] A. Barua, S. W. Thomas, and A. E. Hassan, "What are developers talking about? an analysis of topics and trends in stack overflow," *Empirical Software Engineering*, vol. 19, no. 3, pp. 619–654, 2014.

[29] G. Gousios, "The ghtorent dataset and tool suite," in *Proceedings of the 10th working conference on mining software repositories*. IEEE Press, 2013, pp. 233–236.

[30] S. S. Manes and O. Baysal, "How Often and What Stack Overflow Posts Do Developers Reference in Their GitHub Projects?" in *Proc. of the International Conference on Mining Software Repositories*, 2019, pp. 235–239.

[31] G. Gousios, "The GHTorrent Dataset and Tool Suite," in *Proc. of the Working Conference on Mining Software Repositories*, 2013, pp. 233–236.

[32] W. Muylaert and C. De Roover, "Prevalence of Botched Code Integrations," in *Proc. of the International Conference on Mining Software Repositories*, 2017, pp. 503–506.

[33] M. Beller, G. Gousios, and A. Zaidman, "TravisTorrent: Synthesizing Travis CI and GitHub for Full-Stack Research on Continuous Integration," in *Proc. of the International Conference on Mining Software Repositories*, 2017, pp. 447–450.

[34] K. Werder and S. Brinkkemper, "MEME - Toward a Method for EMotions Extraction from GitHub," in *Proc. of the International Workshop on Emotion Awareness in Software Engineering*, 2018, pp. 20–24.

[35] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian, "The promises and perils of mining github," in *Proceedings of the 11th working conference on mining software repositories*. ACM, 2014, pp. 92–101.

[36] C. Treude and M. Wagner, "Predicting good configurations for github and stack overflow topic models," in *Proceedings of the 16th International Conference on Mining Software Repositories*. IEEE Press, 2019, pp. 84–95.

[37] J. Liao, G. Yang, D. Kavaler, V. Filkov, and P. Devanbu, "Status, identity, and language: A study of issue discussions in github," *PloS one*, vol. 14, no. 6, p. e0215059, 2019.

[38] G. A. A. Prana, C. Treude, F. Thung, T. Atapattu, and D. Lo, "Categorizing the content of github readme files," *Empirical Software Engineering*, vol. 24, no. 3, pp. 1296–1327, 2019.

[39] A. A. Bangash, H. Sahar, S. Chowdhury, A. W. Wong, A. Hindle, and K. Ali, "What do developers know about machine learning: a study of ml discussions on stackoverflow," in *Proceedings of the 16th International Conference on Mining Software Repositories*. IEEE Press, 2019, pp. 260–264.

[40] P. Chatterjee, K. Damevski, L. Pollock, V. Augustine, and N. A. Kraft, "Exploratory study of slack q&a chats as a mining source for software engineering tools," in *Proceedings of the 16th International Conference on Mining Software Repositories*. IEEE Press, 2019, pp. 490–501.

[41] S. A. Chowdhury and A. Hindle, "Mining stackoverflow to filter out off-topic irc discussion," in *Proceedings of the 12th Working Conference on Mining Software Repositories*. IEEE Press, 2015, pp. 422–425.

[42] A. Arwan, S. Rochimah, and R. J. Akbar, "Source code retrieval on stackoverflow using lda," in *2015 3rd International Conference on Information and Communication Technology (ICoICT)*. IEEE, 2015, pp. 295–299.

[43] C. Greco, T. Haden, and K. Damevski, “StackInTheFlow: Behavior-Driven Recommendation System for Stack Overflow Posts,” in *Proc. of the International Conference on Software Engineering*, 2018, pp. 5–8.
[44] J. Zou, L. Xu, W. Guo, M. Yan, D. Yang, and X. Zhang, “Which Non-functional Requirements Do Developers Focus On? An Empirical Study on Stack Overflow Using Topic Analysis,” in *Proc. of the Working Conference on Mining Software Repositories*, 2015, pp. 446–449.
[45] M. Linares-Vásquez, B. Dit, and D. Poshyvanyk, “An Exploratory Analysis of Mobile Development Issues Using Stack Overflow,” in *Proc. of the Working Conference on Mining Software Repositories*, 2013, pp. 93–96.
[46] M. M. Rahman and C. K. Roy, “An Insight into the Unresolved Questions at Stack Overflow,” in *Proc. of the Working Conference on Mining Software Repositories*, 2015, pp. 426–429.
[47] A. Soni and S. Nadi, “Analyzing Comment-Induced Updates on Stack Overflow,” in *Proc. of the International Conference on Mining Software Repositories*, 2019, pp. 220–224.
[48] J. Shao and Y. Sun, “A recommendation service for programming study based on stack overflow,” in *Proc. of the IEEE World Congress on Services*, 2018, pp. 13–14.
[49] H. Yin, Z. Sun, Y. Sun, and W. Jiao, “A Question-Driven Source Code Recommendation Service Based on Stack Overflow,” in *Proc. of the World Congress on Services*, vol. 2642-939X, 2019, pp. 358–359.
[50] S. Thompson, *Sampling*, ser. CourseSmart. Wiley, 2012.
[51] C. Treude and M. Wagner, “Predicting Good Configurations for GitHub and Stack Overflow Topic Models,” in *Proc. of the International Conference on Mining Software Repositories*, 2019, pp. 84–95.
[52] S. Baltes and S. Diehl, “Usage and attribution of stack overflow code snippets in github projects,” *Empirical Softw. Engg.*, vol. 24, no. 3, p. 1259–1295, Jun. 2019.
[53] Q. Xuan, M. Gharehyazie, P. T. Devanbu, and V. Filkov, “Measuring the Effect of Social Communications on Individual Working Rhythms: A Case Study of Open Source Software,” in *Proc. of the International Conference on Social Informatics*, 2012, pp. 78–85.
[54] D. Kalpić, N. Hlupić, and M. Lovrić, *Student’s t-Tests*. Springer Berlin Heidelberg, 2011, pp. 1559–1563.
[55] GitHub Documentation, “Git Basics - Viewing the Commit History,” https://git-scm.com/book/en/v2/Git-Basics-Viewing-the-Commit-History, 2020-03-09, [Online; accessed 28-October-2018].